Velocity: Scalability Improvements in Block Propagation Through Rateless Erasure Coding

Nakul Chawla, Hans Walter Behrens, Darren Tapp, Dragan Boscovic, K. Selçuk Candan
Arizona State University
Tempe, Arizona 85281–3673
{nchawla3, hwb, dtapp, dboscovi, candan}@asu.edu

***Abstract*—Blockchain technology and other distributed ledger systems fill an important role in resilient data storage and publication. However, their normally-decentralized methods also bring unique challenges distinct from more traditional approaches. One lies in the replication of data between participating nodes; since no individual node is more trusted than any other, each node maintains its own copy of the entire ledger for the purposes of validating new transactions. Improving how this information is stored and more importantly, how it propagates across the network, are open research questions. In this work, we propose Velocity, a novel block propagation approach based on fountain codes, allowing for better decentralized delivery of blocks and reduced network bottlenecks without sacrificing the security guarantees of the blockchain ledger itself. We also provide an assessment of economic incentives and their impact on participant behavior, showing that the proposed approach is financially beneficial to rational actors. We conclude by showing experimentally that this approach permits for the mining of even larger blocks, thereby increasing transaction throughput of the system compared to existing state of the art methods.**

***Index Terms*—online banking, data transfer, protocols, cryptography**

I. Introduction

Blockchain technology and distributed ledgers in general have kindled a wildfire of development and research in the literature since their popularization with the Bitcoin protocol [1]. In such systems, small atomic data transactions are committed to a shared data structure, or ledger. Decentralized consensus algorithms ensure consistency of this data, even in the presence of adversarial participants [2].

In blockchains, transactions are committed to the ledger in batches, or blocks, to improve the speed and scalability with which these transactions can be processed. These blocks are then shared with other nodes in the network, and their validity can be checked, using a predefined method, in conjunction with already-committed data.

In this paper, we focus on the propagation process of these blocks throughout the network. In general, the larger the block, the more transactions it can contain; this correlates with increased transaction throughput of the network. However, larger blocks also complicate the propagation process, in which the block is passed from one node to another throughout the network until all nodes have the new information.

These traditional all-or-nothing approaches for providing block propagation, although widely adopted [1], [3], [4], are not suitable when blocks become extremely large. This is because the data is transmitted within the network as a large bolus of information from node to node and, should any network interruptions or other issues arise during the transmission process, that information may need to be sent again. Transmission recovery schemes can mitigate this drawback, but do not usually consider adversarial involvement. In either case, the sharing node is likely to saturate its outbound connection when transmitting and remain idle otherwise.

Despite these drawbacks, such approaches still provide an exponential progression in block propagation; each node can share a block many times, sending it to a new node at each time step. That is, the propagation process can be considered a Maclaurin series with geometric behavior [5], and therefore rapidly propagates the entire network, similar to mechanics of cellular mitosis. However, especially at the early stages of propagation, any disruption of a particular replication step will significantly delay the overall propagation.

We therefore formulate the following research questions:

1) Can we increase block size and transaction throughput while maintaining network behavior?
2) Can a propagation method be chosen that improves resilience to communication disruption?
3) Can the adversarially-resilient nature of existing approaches be maintained under the revised scheme?
4) Is a new propagation approach likely to be adopted by economically-rational actors?

To answer these questions, we propose Velocity, a novel approach for block propagation that leverages fountain codes to provide more fine-grained transmission over lossy channels. We further demonstrate that the proposed scheme satisfies the criteria specified above, and we experimentally show that it offers substantial performance benefits over existing approaches. By supporting increased block sizes without impacting security and consensus guarantees, Velocity offers potentially significant increases in transaction throughput.

The remainder of this work is structured as follows. First, we offer an assessment of the current state of the art and relevant background topics in Section II. We propose our method in Section III, followed by an analysis and evaluation in Sections IV and V. We conclude in Section VI.

II. Related Work

A. Blockchain & Distributed Ledger

Distributed ledgers provide a mechanism for information to be packed into *blocks* and shared among untrusted participants, while allowing participants to come to consensus on the data committed to the ledger (or *chain*); together, this is commonly referred to as a *blockchain* from these terms, introduced by Nakamoto [1]. The process of validating blocks and coming to consensus on the ledger is outside the scope of this work, but can be accomplished using existing validation methods, such as proof-of-work, proof-of-stake, or others [1], [6]. These remain an active area of research, with many enhancements and alternatives proposed in the literature [4], [7].

We highlight two implementations in particular. The Bitcoin protocol [1] is best-known for popularizing the idea of blockchains, and serves as a baseline for desirable resiliency and serviceability characteristics. Dash [3] augments its implementation with semi-trusted nodes and a more privacy-preserving ethos; it also serves as the foundational blockchain technology for this work's evaluation.

A key component of blockchain systems is the method used to propagate the ledger information. This is often accomplished by sending the data directly to nodes which request it through a sequential or round-robin approach. When this applies to new nodes, it is referred to as synchronization (or *bootstrapping*), with several works looking at speeding this process [8], [9]. Additional works have focused on reducing the amount of information required to be housed on each node [10], [11]; this family of approaches are commonly referred to as *sharding*.

When data is exchanged between previously-synchronized nodes, this is referred to as block *propagation*. Several alternatives have been proposed to reduce the amount of data transmitted [12]–[14] during propagation; we now examine these in more detail.

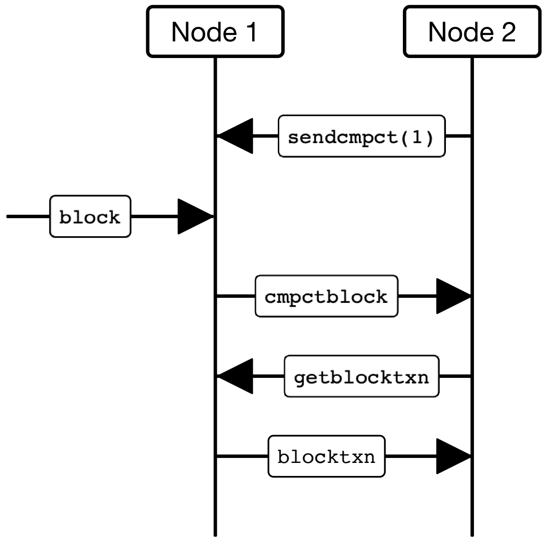

Fig. 1. High-bandwidth Compact Block Propagation

1) Blockchain Propagation: Compact Blocks: Bitcoin Improvement Protocol (BIP) 152 [14] proposed and implemented *compact blocks* as a payload reduction technique. Compact blocks are based on the idea that most transactions on the network have already been transmitted to the peers, and only the set difference must be sent for efficient relay.

This relay process adapts to available bandwidth, and includes high-bandwidth (Fig. 1) and low-bandwidth (Fig. 2) variants, trading excess bandwidth to reduce round-trip time (RTT).

This approach provides meaningful reductions in payload size, a key advantage for accelerating the propagation process. However, it requires the transmission of "short IDs" to identify the missing transactions, which take additional space. Additionally, the transmission process of unknown transactions is not directly optimized, potentially allowing bottlenecks to form when a single node has processed many (otherwise unknown) transactions.

2) Blockchain Propagation: Xtreme Thinblocks: A competing approach was proposed in Bitcoin Unlimited Improvement Protocol (BUIP010) [12]. This approach, called Xtreme Thinblocks, or XThin blocks, leverages Bloom filters for more efficient set comparisons, as seen in Fig. 3.

This approach permits more efficient detection of missing transactions through these filters, but the transmission of this encoded membership information as part of the propagation process adds overhead. Malicious nodes can also send falsified filters to create collision attacks. Finally, additional back-and-forth communication between nodes can delay propagation for high-latency networks.

3) Blockchain Propagation: Graphene: To address the drawbacks of XThin blocks, a followup was proposed in

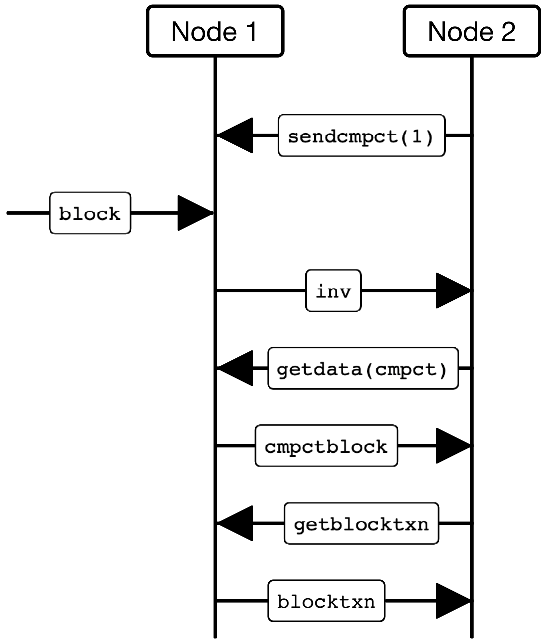

Fig. 2. Low-bandwidth Compact Block Propagation

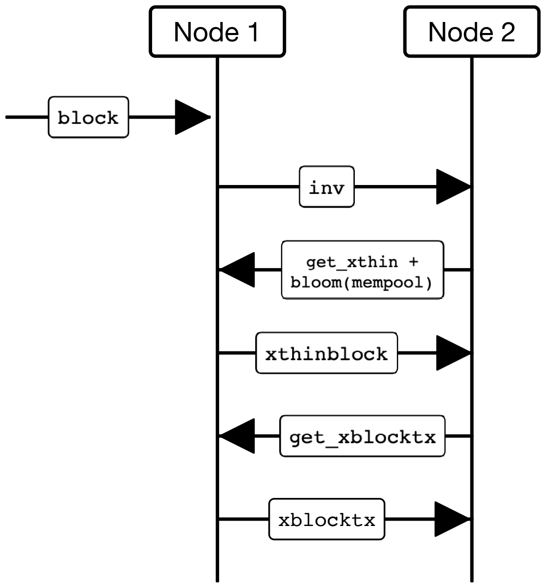

Fig. 3. Xtreme Thinblocks Propagation

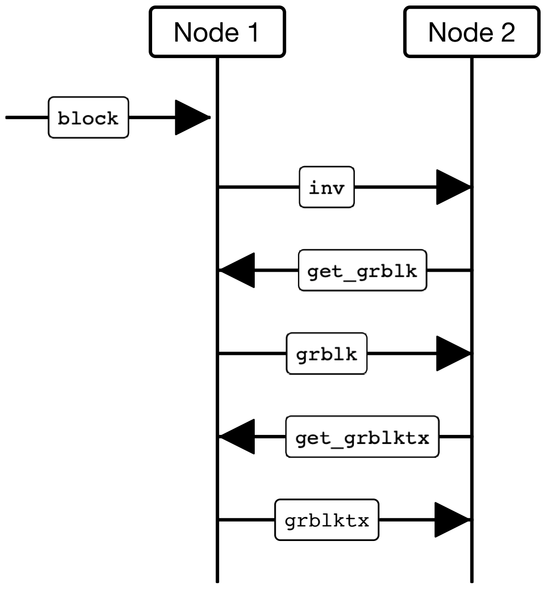

Fig. 4. Graphene Block Propagation

Bitcoin Unlimited Improvement Protocol (BUIP093) [13], commonly called Graphene. Shown in Fig. 4, this approach uses Invertible Bloom lookup tables (IBLT) and compression to improve propagation efficiency even more by further decreasing payload size.

Although computationally more expensive, decreases in bandwidth utilization more than offset the additional cost, since nodes are rarely computationally-bound. However, this approach still relies on the nodes maintaining an in-memory pool (or *mempool*) of transactions, making it impractical for initial node bootstrapping, since such nodes are unaware of any transactions.

4) Blockchain Propagation: Summary: These techniques (a) rely on memory pool synchronization of nodes communicating the blocks and re-transmission of complete blocks on receive failures. Moreover, (b) each of these protocols rely on a single peer to send the complete data as opposed to using multiple peers to transmit partial data, and (c) they can only be used to receive blocks that are currently being propagated, and cannot be used for node bootstrapping. In Section III, we propose a novel protocol to address key deficiencies of these existing protocols.

B. Decentralized Data Publication

One approach adopted to facilitate distributed data publication is the use of distributed hash tables (DHTs) to enable peer-to-peer discovery. For example, Kademlia [15] is used by BitTorrent clients to find other peers with which to directly exchange data. In this system, a bootstrapping process is used to share a common DHT among the participants, with one node responsible for its construction (the initial seeder).

This approach effectively addresses the bottleneck issue by allowing nodes that have only partial information to serve as new sources of those pieces. It is therefore used for peer discovery in some blockchain systems [16]. However, this system faces two downsides that make it challenging to apply. First, the incentives for cooperative participation and protocol adherence can be somewhat distorted by the competitive environment fostered by proof-of-work mining. Second, these systems often require a semi-trusted central node for bootstrapping discovery; this is not a commonly-accepted assumption under most blockchain systems, which generally aspire to true decentralization.

C. Fountain Codes

Fountain codes, also known as rateless erasure codes [17], provide a mechanism by which information can be encoded such that the resulting segments can be probabilistically re-assembled into the original data when the volume of received segments exceeds a threshold. Since reconstruction is based on volume and not on specific blocks, it provides more robust transmission and reassembly behavior compared to DHT-supported direct approaches described above. Shokrollahi proposed additional enhancements to decrease computational load [18], improving transcoding efficiency further. Fountain codes have been used in cooperative data transmission and aggregation applications [19], [20], but have not previously seen usage in the blockchain domain.

Several fountain coding implementations exist, and follow a similar functional paradigm. A piece of information is split up into equally-sized fragments, which are then encoded into *symbols* using an implementation-specific approach. These symbols are transmitted from one or more sources to a recipient, who aggregates this information and inputs the set a decoding algorithm. The likelihood of decoding a set of input symbols into the original data is near zero before k symbols have been received, and approaches one with $(k+\epsilon)$ symbols; ϵ is implementation-specific, and generally less than 2% of k for a near-certain likelihood of reconstruction.

III. Velocity: Breaking the Block Size Barrier

To address these challenges, we introduce Velocity, which uses the error correcting properties of rateless erasure codes to allow multi-origin, simultaneous-broadcast block propagation, rather than single-source direct transmission. The high-level operation of the Velocity protocol (see Fig. 5) is as follows:

1) A sender node, on hearing about a new block[1], sends an `inv` to neighboring peers. The block body is reinterpreted as a byte array and encoded into a set of fixed-length symbols with a pre-negotiated size.
2) On receiving an `inv` message, a receiver requests any unknown blocks using a `get_sym` request to all its neighboring peers, including the originating node.
3) Peers who have information on the requested block(s) respond with repeated `sym` responses, each encoding one symbol. Nodes which cannot contribute any missing blocks ignore the `get_sym` request.
4) The receiver collects these symbols in a singleton class containing a map from block header IDs to collections of symbol pairs. In order to prevent DoS attacks, every symbol must contain header information. In smaller blocks, this can be optimized to alternatively send only the 64-byte header hash instead.
5) Receivers know the quantity of symbols needed to reconstruct the block with high likelihood, and wait until they receive the target volume of symbols.
6) Upon reaching the required count, a receiver reconstructs that block; if reconstruction fails, it can either wait for more symbols, or attempt to discover incompatible symbol pairs (which would indicate adversarial behavior).
7) If reconstruction succeeds, a `received` message is sent to the responding peers to stop symbol transmission.

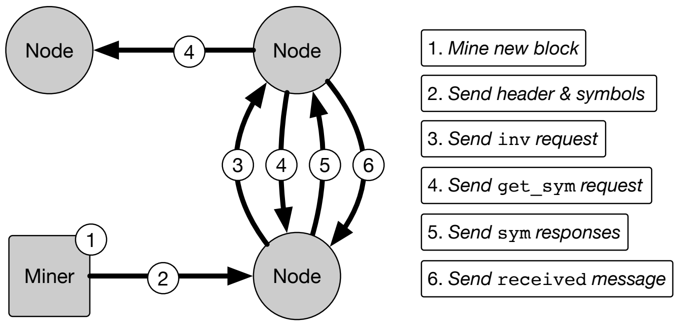

Fig. 5. An example of the protocol's communication flow

To support this protocol, we have introduced three new messages: `get_sym`, which requests the encoded symbols of missing blocks from peers, `sym`, which contains a single encoded symbol corresponding to a requested block, and `received`, which notifies nodes to stop the flow of `sym` replies for that block. `get_sym` requests are sent to all connected peers when an `inv` request informs a participant about a previously-unknown block.

[1]Discovery can occur directly from a miner, from a peer via an `inv` request, or through an alternative pull-based method (if available).

A. Adversarial Resilience

Thus far, our discussion has been limited to behavior where the participants adhere to the protocol details. However, given the financial incentives involved in blockchain-related attacks, we also consider cases where nodes within the network subvert or ignore intended protocol behavior. To this end, we address several possible adversarial behaviors.[2]

First, a node may send an `inv` request containing non-existent blocks, causing the recipient node(s) to send superfluous `get_sym` requests to their neighbors. However, since these requests will be ignored when the block does not exist, the total number of messages generated will be low. Additionally, repeated "dud" requests of this type from a single node would be easy to detect, leading to a blacklisting of the misbehaving node.

Second, a node may send a `get_sym` request for a block it already has, and then withhold the corresponding `received` message. This ties up the uplink bandwidth of the targeted node in sending superfluous `sym` responses. This can be addressed by limiting the number of symbols sent to a given node, for a specified block, to a certain number equal to the quantity needed to reassemble the source block $(k + \epsilon)$, plus a surplus sufficient to compensate for symbols lost in transmission. The number of such "timed-out" blocks can then be tracked on a per-node basis as above, with repeated offenders again being blacklisted.

Finally, an adversarial node may correctly respond with `sym` to a valid node's `get_sym` request; however, the symbol itself may be improperly constructed or totally falsified. This will greatly complicate the reconstruction process on the recipient node, and potentially make it impossible. To address this challenge, each node must maintain a two-dimensional matrix of node-symbol pairs; during the decode process, if a given node has provided a symbol that conflicts with a previously-reconstructed chunk, the conflicting nodes are flagged and later symbols from them can be scrutinized through a pairwise ensemble with unflagged nodes, or ignored. Maintaining this data requires additional RAM, but usage will be constant as a function of neighbors and block size, a reasonable tradeoff.

IV. Analysis: Block Size vs Revenue

A. Economic Incentives

Even if a distributed ledger can maintain consensus with larger blocks, it may happen that miners will choose to mine smaller blocks, or *pre-mine*. If we assume that miners are rational actors and intend to maximize revenue, we can analyze the economic conditions that will guide miners' decisions to include more transactions in a block.

In practice, mining pools often try to optimize their block size for revenue using historical data. With some simplifying assumptions, we can identify a point where miners may decide that the fee that a transaction carries does not justify the higher

[2]Inter-node communication is cryptographically signed, ensuring the source of a given message cannot be spoofed; this eliminates many classes of attacks.

risk of orphaning the (larger) block. We define the following variables for our analysis:

1) R: The mining reward for the block
2) F: The fee density in Dash/MB
3) x: The block size in MB
4) $\phi(x)$: The probability of mining an orphan block as a function of the block size

Let us initially assume that the fee density of transactions, that is, the fee per MB, is constant. In such a case, the expected revenue (ER) is provided by

$$ER = (R + Fx)(1 - \phi(x)).$$

To maximize this function we take the derivative with respect to block size and set it equal to zero.

$$0 = F(1 - \phi(x)) - (R + Fx)\phi'(x)$$

If the orphan rate increases linearly with block size, $\phi(x) = \beta x$, $\beta > 0$. Then we may solve

$$0 = F - \beta F x - R\beta - \beta F x$$

or

$$x = \frac{F - R\beta}{2\beta F}$$

With these assumptions, we can determine the optimal block size to maximize the revenue of miners. Note that the optimal size decreases with the mining reward; in the absence of a mining reward optimal block size is only determined by β.

In the case of transactions of non-homogeneous fee density, it seems reasonable that optimal block size would decrease. This follows from the observation that transactions with high fees would act as a higher block reward when miners include them (with high priority) in blocks.

Using data from our simulation results in Section V, we perform a linear regression on block sizes less than 500MB, assuming that the orphan rate is zero for blocks with no transactions. Note that this assumption is not guaranteed; additional discussion on this point follows in the next subsection.

This gives an approximate β of 0.000665. Currently the Dash network has a mining reward of 1.6728 Dash, and the minimum relay fee requires fee density to be larger than 0.01 Dash/MB; of this, 0.005 Dash/MB is earned by the miner. Using these values, we find that miners would maximize revenue around blocks of size 292MB.

The coefficient of determination of our linear regression is 0.938, which does suggest a linear correlation. However, more data points would allow a more thorough understanding of the relationship between blocksize and orphan rate.

B. Subversive Miners

The above analysis assumes straightforward, economically rational miners. However, financial incentives for adversarial behavior can distort this assumption unexpectedly. Previous measurements have shown that orphan rates in other networks do not reach zero [21]; additionally, miners sometimes disregard fee density and mine empty blocks. In cases of low transaction load, or when mining reward and fee density are not correctly balanced, this behavior produces higher earnings while failing to provide meaningful benefit to the underlying network.

Other types of collusion, such as selfish mining, have previously been found to be economically rational as well [22]. Previous works [23] have addressed some of these challenges, but these operate using information associated with the blocks, rather than the block propagation behavior directly. Correct block propagation may be incentivized [24], or incorrect propagation may be punished [25], but the data delivery mechanics of these schemes are not involved. Therefore, due to its modular nature, Velocity could be integrated with existing mitigation approaches that do not rely on block propagation as their protection mechanism.

As such, an economic evaluation restricted to non-adversarial miners is not incompatible with practical deployments. Rather, it can be considered in conjunction with existing systems to discourage these behaviors, where the only option for rational agents is protocol compliance. Under these assumptions, block size can be considered disjoint from the problem of subversive behavior relative to block mining and propagation.

V. Performance Evaluation

A. Simulator Configuration

In order to understand the scalability characteristics of Velocity, we ran experiments with varying block sizes, symbol counts, and numbers of connections. Dash provides a local-only regression test mode, but since this mode does not consider latency between nodes, it is not applicable for our performance evaluation. Dash's testnet, a fully-functional blockchain network designed for testing, is also not suitable for large block size testing due to very low transaction traffic.

In [26], we developed a comprehensive, robust, and realistic blockchain simulator for the Dash network. In particular, we have extended the simulation framework previously presented in [27], which is based on Network Simulator 3 (NS3) [28], and was originally developed to illustrate Bitcoin, Dogecoin, and Litecoin systems. [27] treated blocks as a fundamental component; we extended the implementation to provide a more robust representation of block propagation. We also added support for compact block, Xtreme Thinblock, and Velocity propagation, for comparison purposes.

The network is initialized randomly, with nodes assigned roles as full nodes, master nodes[3], or miners with equal likelihood, matching the real-world distribution of the Dash network. Miner nodes are fully connected with each other, reflecting a real-world topological observation that miners maintain private peering connections among themselves [29]. Connections between other neighboring nodes are established as point-to-point channels, serving to abstract the underlying architectures (such as TCP/IP) that form these links. Each link

[3]Master nodes and full nodes are differentiated primarily by their bandwidth, reflecting the real-world characteristics of the Dash network.

is assigned two characteristics, a latency L and a bandwidth B. Finally, each node maintains its own mempool, or cache of transactions, to replicate real transaction processing techniques [3].

For symbol encoding and decoding, we used the LibRaptor library [30]; more generally, the fountain code encoding and decoding method used by Velocity can be considered to be implementation-agnostic, with a fixed computational overhead proportional to the quantity of data to encode. As propagation time dominates encoding and decoding times, this provides a better evaluation of the propagation process itself, and the advantages that Velocity offers without requiring a particular transcoding implementation.

The simulator's source code is publicly available at [26].

B. Experimental Parameters

In addition to the network topology, described above, each simulation varies the following parameters:

- S: The block size, in MB
  Varies from 100MB to 1GB
- I: The block interval of the network, in seconds
  $I = 150$*, taken from the Dash protocol*
- P: The propagation methodology chosen
  Traditional or Velocity
- N: The number of blocks to propagate (i.e. duration)
  $N = 100$*, equivalent to* 15000 *seconds*
- k: The number of fountain code symbols per block
  Varies from 10 to 10000

To model realistic transaction loads, we use a Gaussian distribution across nodes, with each node generating transactions at a rate sampled from a piecewise constant distribution based on real-world transaction data collected directly from the Dash ledger. Randomly-generated transaction workloads indirectly cause stochastic network routing and load behavior, while also maintaining characteristics similar to real-world load observations. Additionally, blocks are generated using geometric progressions from random seeds, which in turn results in blocks that may be generated at different nodes for runs with similar parametric configurations.

C. Results

We evaluated the approaches using the following criteria:

- Orphan rate: The orphan rate tells us the percentage of mined blocks which failed to be added to the blockchain.
- Median propagation time: The median time cost (in seconds) for block propagation to all of the nodes. If this exceeds I, the network will *fall behind*, as new blocks will be produced faster than they can propagate.

1) Choosing the symbol size: In order to aggregate the symbols from the peer nodes, we need to understand what the ratio between block size and symbol size should be in order for the propagation to be optimal. The following observations are based on Fig. 6.

- For $k = 10$, we see that for 100MB blocks, the orphan rate starts at a (relatively) low value of 9%, but it increases quickly with block size. This is because the symbol size begins to saturate the upload bandwidth for the node, and therefore takes additional time to propagate.
- For $k = 100$ and $k = 1000$, the orphan rate increases at almost the same rate. It is notable that the resulting symbols are relatively small, and therefore no individual symbol monopolizes the outbound bandwidth of the node. This reduces propagation time, thereby also reducing orphaned blocks.
- For $k = 10000$, smaller block sizes (between 100MB and 400MB) performance is acceptable. As block size grows, the orphan rate becomes similar to that of $k = 100$ and $k = 1000$, although still significantly lower than the rates under non-symbolic block propagation.

2) Orphan Rates: In addition to the above conclusions, Fig. 6 also supports the following observations regarding the orphan rates:

- Under the non-symbolic propagation scheme, the orphan rate approaches 97%; that is, almost all blocks produced are orphaned when using 1GB blocks. Additionally, for all block sizes between 100MB and 1GB, the network is unable to maintain consensus, establishing an upper-bound on (usable) block size under the original scheme.
- In contrast, we see that when the data is aggregated from multiple peers, the orphan rate is contained below 50% for 400MB blocks with $k = 10$, and for k between 100 and 10000, the network maintains consensus below block sizes of 600MB.

3) Block Propagation Time: From Fig. 7 we observe the following:

- The median block propagation time cost for the original, non-symbolic approaches exceeds I with block sizes of 200MB. It is especially noticeable from Fig. 6 that 200MB blocks also do not reach consensus under the original approach.
- The median block propagation time remains below I using Velocity propagation for block sizes up to 400MB for $k = 10$, and up to 600MB for k between 100 and 10000. Furthermore, Fig. 6 shows that consensus is maintained for block sizes up to 400MB for $k = 10$, and for block sizes of up to 600MB with k values between 100 and 10000.

Fig. 8 summarizes the relationship between orphan rate and median propagation time. In Fig. 8(a), we plot the relationship between these two parameters for two different per-block symbol counts, $k = 10$ and $k = 10000$. In the figure, each dot corresponds to a different block size. As we see in the figure, the median propagation time is exponentially proportional to the orphan rate; moreover, both the proportionality constant and the exponent is higher for $k = 10$ than $k = 10000$. Consequently, $k = 10000$ leads to overall lower orphan rates and median propagation times than $k = 10$. Note that the figure also shows that, as expected, both orphan rates and propagation times increase with the block size. Fig. 8(b) plots the relationship between orphan rate and median propagation

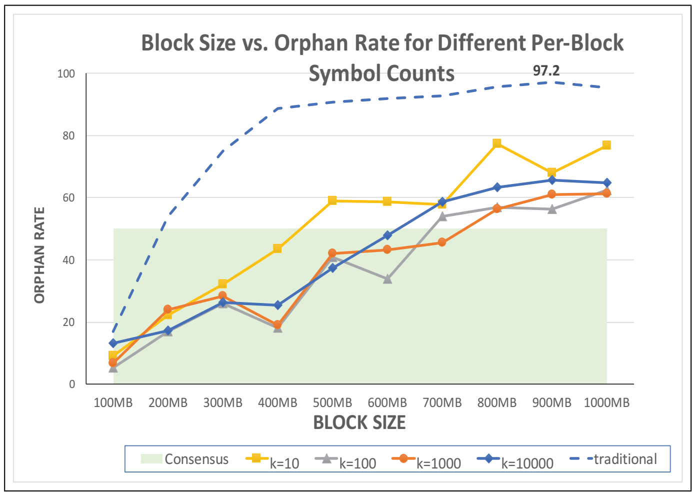

Fig. 6. Block size vs. orphan rate for different per-block symbol counts (k)

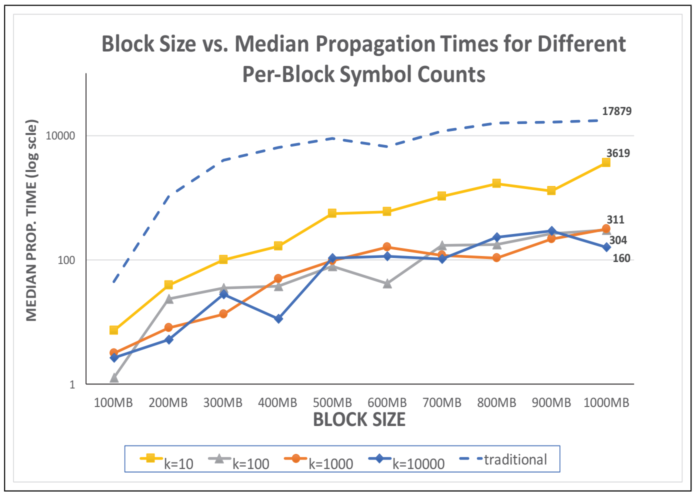

Fig. 7. Block size vs. propagation time for different per-block symbol counts (k)

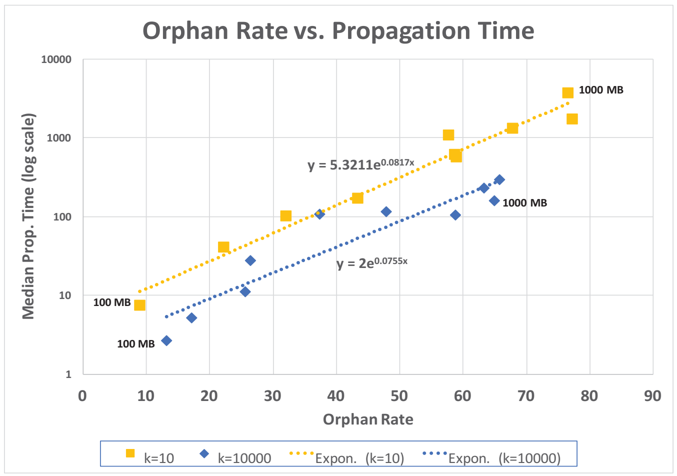

Fig. 8(a) different per-block symbol counts (k)

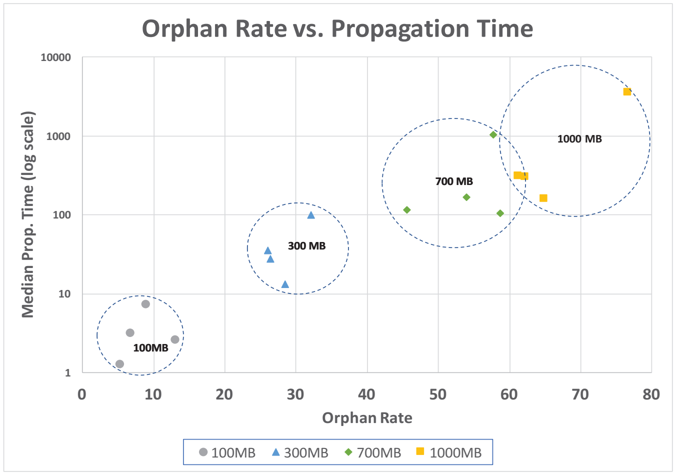

Fig. 8(b) different block sizes

Fig. 8. Orphan rate vs. propagation time for (a) different per-block symbol counts, k, and (b) different block sizes

rate for different block sizes (100MB, 200MB, 700MB, and 1000MB); each point in a region corresponds to a different per-block symbol count, k. This confirms the observation that both orphan rates and median propagation times increase with block size.

D. Discussion & Limitations

Previously, [31] found that the Dash network can scale to 4MB blocks using the 'compact' propagation approach, or to 10MB blocks using the 'XThin' approach, while still maintaining orphan rates below 1%. As mentioned in Section IV, these rates play an important role in generating realistic actor self-interest modeling. Our results have shown that using the proposed Velocity protocol as part of hybrid (compression-enabled) propagation strategy can facilitate increases in block size (and indirectly, transaction throughput) for a network which adopts this approach.

We note that while orphan rates are somewhat higher under Velocity than non-symbolic approaches, their levels must be capped below a threshold at which the consensus of the network will break down. Although these higher rates increase both miner profitability and transaction throughput, they also negatively impact computational efficiency and network latency, which are nevertheless key performance metrics for blockchain systems [32], [33]. To mitigate these drawbacks, "compromise" orphan rates could be discovered through further block size tuning, balancing these competing requirements prior to real-world deployment.

VI. Conclusions & Future Work

Current propagation techniques result in fast propagation and low orphan rates for small blocks, and are highly dependent on memory pool synchronization. If memory pools (of the peers propagating blocks) do not synchronize, the methods fail to relay blocks faster, and fall back on traditional propagation.

Analyzing the above results, we can conclude that utilizing data aggregation techniques on a node for data propagation works better than syncing block using one peer. It must be noted that full nodes are personal machines that do not have high bandwidth. They get several advantages when relaying symbols instead of the full blocks:

- Miners have the capability of multi-casting. In our experiments, we have used simultaneously-scheduled uploads over TCP connections. These may serve large blocks to many nodes using continuous asynchronous streams, which is not possible in traditional relay approaches.
- The data received by a full node is downloaded through multiple peers in parallel, resulting in faster downloads.

The above advantages help nodes to propagate large blocks faster. We have seen that the orphan rate improves when sending fountain codes instead of fetching the block from a single client; for example, the rate for 100 MB blocks drops from 17% to 9%, increasing miner profitability. We conclude that the network can maintain consensus up to 600MB blocks, compared to 100MB for conventional approaches.

We finally note that the improvements offered by Compact, XThin, and Graphene block propagation approaches are not mutually incompatible with Velocity. By significantly reducing the payload while sending a draft or summary of the whole block, orphan rates can be reduced while transmission throughput is increased. We intend to explore the extension of these approaches through a hybrid method that leverages the advantages of fountain codes to improve the re-transmission of block data by these methods (`blocktxn`, `xblocktx` and `grblocktx`, respectively). However, significant challenges remain in harmonizing the encoding and decoding requirements with mempool requirements on each node.

Acknowledgment

We would like to offer our sincere thanks to the Dash community for their kind support of this research, which was funded in part by Dash Treasury Grant ASU-BC-LAB-201707. We would also like to gratefully acknowledge our reviewers for their thoughtful and attentive feedback, and the Dash team for sharing their expertise throughout the course of this work.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] K. Wüst, "Security of blockchain technologies," Master's thesis, ETH Zürich, 2016.

[3] E. Duffield and D. Diaz, "Dash: A privacycentric cryptocurrency," *Self-published*, 2015.

[4] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform, 2013," *URL* {*http://ethereum. org/ethereum. html*}, 2017.

[5] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55. Courier Corporation, 1965.

[6] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357–388, Springer, 2017.

[7] C. Cachin and M. Vukolić, "Blockchains consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[8] D. Leung, A. Suhl, Y. Gilad, and N. Zeldovich, "Vault: Fast bootstrapping for the algorand cryptocurrency," in *NDSS*, 2019.

[9] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 931–948, ACM, 2018.

[10] A. E. Gencer, R. van Renesse, and E. G. Sirer, "Short paper: Service-oriented sharding for blockchains," in *International Conference on Financial Cryptography and Data Security*, pp. 393–401, Springer, 2017.

[11] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, ACM, 2016.

[12] P. Tschipper, "Buip010: Xtreme thinblocks," January 2016. https://github.com/BitcoinUnlimited/BUIP/blob/master/010.mediawiki.

[13] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. Levine, "Graphene: A new protocol for block propagation using set reconciliation," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 420–428, Springer, November 2017.

[14] M. Corralo, "Bip152: Compact block relay," April 2016. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki.

[15] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*, pp. 53–65, Springer, 2002.

[16] L. Anderson, R. Holz, A. Ponomarev, P. Rimba, and I. Weber, "New kids on the block: an analysis of modern blockchains," *arXiv preprint arXiv:1606.06530*, 2016.

[17] M. Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, p. 271–280, Nov 2002.

[18] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2551–2567, 2006.

[19] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4, pp. 2235–2245, IEEE, 2005.

[20] T. Locher, S. Schmid, and R. Wattenhofer, "Rescuing tit-for-tat with source coding," in *Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)*, pp. 3–10, IEEE, 2007.

[21] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," *International Conference on Financial Cryptography and Data Security*, 2018.

[22] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[23] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pp. 45–59, 2016.

[24] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On bitcoin and red balloons," in *Proceedings of the 13th ACM conference on electronic commerce*, pp. 56–73, ACM, 2012.

[25] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 279–296, 2016.

[26] N. Chawla, "Dash simulator." Github https://github.com/thenakulchawla/dash-simulator.git, 2017. Accessed : 2018-10-20.

[27] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3–16, ACM, 2016.

[28] G. Carneiro, "Ns-3: Network simulator 3," in *UTM Lab Meeting April*, vol. 20, 2010.

[29] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 375–392, IEEE, 2017.

[30] L. Fulchir, "libraptorq." https://github.com/LucaFulchir/libRaptorQ.

[31] D. Boscovic, N. Chawla, and D. Tapp, "Block propagation applied to nakamoto networks," 2018.

[32] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," 2014.

[33] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 253–256, IEEE, 2017.